Paper 007-2007

The FILENAME Statement Revisited

Yves DeGuire, Statistics Canada, Ottawa, Ontario, Canada

ABSTRACT

The FILENAME statement has been around for a long time for connecting external files with SAS®. Over the years, it has grown quite a bit to encompass capabilities that go beyond the simple sequential file. Most SAS programmers have used the FILENAME statement to read/write external files stored on disk. However, few programmers have used it to access devices other than disk or to interface with different access methods. This paper focuses on some of those capabilities by showing you some interesting devices and access methods supported by the FILENAME statement.

INTRODUCTION – FILENAME STATEMENT 101

The FILENAME statement is a declarative statement that is used to associate a fileref with a physical file or a device. A fileref is a logical name that can be used subsequently in lieu of a physical name. The syntax for the FILENAME statement is as follows:

```
FILENAME fileref <device-type> <'external-file'> <options>;
```

This is the basic syntax to associate an external file or a device to a fileref. There are other forms that allow you to list or clear filerefs. Please refer to SAS OnlineDoc® documentation for a complete syntax of the FILENAME statement.

Once the association has been made between a physical file and a fileref, the fileref can be used in as many DATA steps as necessary without having to re-run a FILENAME statement. Within a DATA step, the use of an INFILE or a FILE statement allows the programmer to determine which file to read or write using an INPUT or a PUT statement.

It is possible to make do without a fileref, but this practice is generally discouraged as it embeds physical names in SAS programs. It is much better to isolate these using FILENAME statements because physical names tend to change over time and are typically non-portable across platforms.

The following example illustrates a typical scenario in a Windows environment:

```
/*************************************************************/
/* Example of a FILENAME statement with the DISK option.     */
/*************************************************************/
%let SASROOT=c:\Program Files\SAS\SAS 9.1; /* location of SAS software */
filename copyin disk "&SASROOT\core\sample\contents.sas";
filename copyout disk "C:\temp\contents.sas";

data _null_;
    infile copyin sharebuffers;
    file copyout;
    input;
    put _infile_;
run;
```

In this example, we associate two filerefs, namely copyin and copyout. Both are associated with an external file on disk (in fact the local C: drive) hence the use of the DISK option which is not necessary since this is the default. The DATA step that follows copies the data from copyin to copyout. The INFILE statement is used to identify copyin as the input file and conversely the FILE statement identifies the output file as copyout. The SHAREBUFFERS option is used to minimize data movement in memory but is not required for this example to work. The INPUT and the PUT statements trigger the I/O and because we want to read all the data without creating any SAS variables, we use the INPUT statement with no argument. The PUT statement writes the data to copyout using the special SAS variable _infile_ which represents the input record.

BEYOND THE DISK DEVICE

Most programmers have used the FILENAME statement to refer to an external file on disk. It is possible to go beyond by specifying a device type or an access method other than disk. (Note that there is a distinction to be made between a device and an access method. A device is a physical computer component whereas an access method can be defined as an interface to a storage medium.)

When using a device other than disk, you have to think of a stream of data being input to your program or being output by your program. The nice thing about this model is that whether you are using a sequential file on disk or another device, the DATA step operates the same way using the INFILE/FILE and the INPUT/PUT statements.

Although the same model applies to all devices, you must be aware that some options are device and platform specific. Furthermore, the devices supported by SAS vary from platform to platform. Make sure to read the documentation pertaining to the operating system under which you intend to run your SAS program. All the devices and examples presented in this paper are based on the Windows operating system. It is possible, however, that some examples presented in this paper apply as is to other operating systems but we urge you to check that it is the case rather than assuming so.

THE TEMP DEVICE

The TEMP device allows you to create a temporary file that exists only as long as the filename is assigned. There are many reasons why you would need to use a temporary file in a SAS program. A good example would be a SAS program that first generates some SAS code that needs to be included and executed later in the same SAS session. If you don't care about making the generated program persistent between sessions, then the generated code should be stored in a temporary file. The following example illustrates this process.

```
/*************************************************************/
/* Example of a FILENAME statement with the TEMP option.     */
/*************************************************************/
filename pgm temp;

/* Create the generated program */
data _null_;
    file pgm;
    put 'data _null_;';
    put '  putlog "I am a generated program;";';
    put 'run;';
run;

/* Run it */
%include pgm;
```

When you use the TEMP device, you do not specify an external file (in the background, SAS generates one for you). This is a nice feature because you don't need to come up with a name that you don't really care about. As well, SAS will make sure that the temporary file gets deleted when the fileref is disassociated either with an explicit FILENAME CLEAR statement or by terminating the SAS session.

THE DUMMY DEVICE

The DUMMY device can be used to provide as input an empty file or to discard an output file. It is useful in the early stages of testing when you are validating the program flow but you are not yet concerned about the actual data processing. It can also be used to provide a mandatory fileref as an argument to a macro (or some other module) when you don't have the required file or when you don't care about a particular output file being created.

In the example below, we have decided to discard the log with the help of the PRINTTO procedure and a DUMMY device. The PRINTTO procedure allows you to redirect the log (and also procedure output) to a file and, by providing the procedure with a fileref that is associated with a DUMMY device, we are capable of discarding the log.

```
/*************************************************************/
/* Example of a FILENAME statement with the DUMMY option.    */
/*************************************************************/
filename outlog dummy;

/* Redirect the log to the DUMMY device so that it is discarded */
proc printto log=outlog;
run;

/* A data step that writes to the log. */
data _null_;
  putlog 'I will disappear!';
run;

/* Reset the log */
proc printto;
run;
```

THE CLIPBOARD ACCESS METHOD

The CLIPBOARD access method allows SAS to read data that have been copied into the Windows clipboard or to write information to the Windows clipboard that can later be pasted into another Windows application. This facility is useful when SAS is run interactively with other Windows applications (Word or Excel for example). It allows sharing information using the traditional cut-and-paste operation.

The next example has been taken from the SAS OnlineDoc documentation and adapted to use the CLIPBOARD access method. It shows the use of the CLIPBOARD access method to create a self-contained program that demonstrates the call of a WIN32 routine. To do so, the SAS MODULEN CALL routine is required along with an external file that describes the WIN32 routine we wish to call (in our case the Beep routine). The external file must be associated with the fileref sastblc. Because we want our example to be self-contained, we use the CLIPBOARD option and instruct the user to copy the lines provided as part of a comment into the Windows clipboard.

```
/*************************************************************/
/* Example of a FILENAME statement with the CLIPBRD option.  */
/*************************************************************/
filename sascbtbl clipbrd;

/* Before submitting: copy the next 8 lines below into the clipboard
routine Beep
  minarg=2
  maxarg=2
  stackpop=called
  callseq=byvalue
  module=kernel32;
arg 1 num format=pib4.;
arg 2 num format=pib4.;
copy the 8 lines above into the clipboard */

/* Run the Beep Windows routine */
data _null_;
  rc = modulen("*e","Beep",1380,1000);
run;
```

THE PIPE DEVICE

The PIPE device allows SAS to capture the output of an external command or to write data for consumption by an external command. The only requirement for the communication to take place is for the external command to write data on standard output (STDOUT) and to read data on standard input (STDIN). STDIN and STDOUT are two special file handles that are always open and that can be used to chain commands one after the other. There is a third special file called standard error (or STDERR) and is used to trap error messages. Also, the PIPE device must not be confused with the NAMEPIPE device which is another option available with the FILENAME statement. The

NAMEPIPE device is much more advanced and allows bidirectional communication between applications on the same computer or across a network. The NAMEPIPE device is not discussed in this paper.

The example below runs the DIR command and dumps its output to the SAS log. In effect, the DIR command lists the files contained in the current directory when SAS opens the fileref curdir.

```
/*************************************************************/
/* Example of a FILENAME statement with the PIPE option.     */
/*************************************************************/
filename curdir pipe "DIR";

data _null_;
  infile curdir;
  input;
  putlog _infile_;
run;
```

THE CATALOG ACCESS METHOD

SAS catalogs are special SAS files that store different kinds of information in smaller units called catalog entries. Each entry has a type that identifies its purpose to SAS. A single SAS catalog can contain several different types of catalog entries. The CATALOG access method allows a SAS program to read a catalog entry or to write directly to a catalog entry.

Some uses of this facility may include the storage of generated code that needs to persist from one SAS session to another (contrary to the example provided with the TEMP device which is kept only for the duration of a SAS session). It can also be used to store parameter data into a catalog entry which will be referenced later via a fileref. This is illustrated in the example below which achieves the same functionality as the CLIPBOARD example but in this case the Beep routine definition is nicely kept in a catalog as a source entry. The example below shows how to create the source entry, but if the catalog is permanent this only needs to be done once. Note as well that you can create a source entry interactively by using the SAS windowing environment.

```
/*************************************************************/
/* Example of a FILENAME statement with the CATALOG option.  */
/*************************************************************/
filename sascbtbl catalog 'work.mycat.beep.source';

/* Create the beep routine definition into a source catalog entry */
data _null_;
  file sascbtbl;
  put 'routine Beep';
  put 'minarg=2';
  put 'maxarg=2';
  put 'stackpop=called';
  put 'callseq=byvalue';
  put 'module=kernel32;';
  put 'arg 1 num format=pib4.;';
  put 'arg 2 num format=pib4.;';
run;

/* Run the Beep Windows routine */
data _null_;
  rc = modulen("*e","Beep",1380,1000);
run;
```

THE EMAIL ACCESS METHOD

From time to time, you would like to notify certain people that a given SAS job has terminated (normally or abnormally) or you would like to send the output produced by a SAS program to people who require the information. You can do that programmatically using the EMAIL access method. Contrary to the other devices we've shown so far, you will need first to set the appropriate SAS email options because SAS needs to know what email system and what email account to use. Although those options can be specified on the FILENAME statement, it is probably more appropriate to save the setup in your SAS configuration file.

Once your email configuration is done, you can start using the EMAIL access method but you will need to pay attention to several options pertaining to a specific message. The destination address is mandatory and can be provided with the TO option. You should also provide a subject line using the SUBJECT option. If you have an attachment, you must use the ATTACH option. In any of these options, you can provide a list enclosed in parentheses and not just a single element. The body of the message is provided by writing to the file using PUT statements in a DATA step.

In the next example, we use email to notify someone of the outcome of the FREQ procedure. We essentially provide the recipient with the return code of the procedure and the output, saved in a SAS data set, is attached to the message. Note that using a macro with conditional macro statements would allow customizing the email message depending whether or not the run was successful. Indeed, there is no need to send the output when the FREQ procedure ends abnormally!

```
/*************************************************************/
/* Example of a FILENAME statement with the EMAIL option.    */
/*************************************************************/
filename msgout email to="email-address";

/* Run Proc Freq */
proc freq data=sashelp.air;
  tables date/out=freqres;
  run;

/* set location of the results to be emailed*/
%let fname=%sysfunc(pathname(work))\freqres.sas7bdat;

/* Email the results */
data _null_;
  file msgout subject="SAS Job terminated"
              attach="&fname";
  put "The SAS job has terminated.";
  put "The resulting report is attached.";
run;
```

THE FTP ACCESS METHOD

The FTP access method implements the File Transfer Protocol (FTP) to access remote files. The traditional FTP commands can be provided as options on the FILENAME statement itself but they are run only when the fileref is opened via an INFILE or a FILE statement. So, anytime one needs to access a file stored on a remote server but no direct facility is provided, the FTP access method should be considered since it makes the whole process seamless. In the following example, we access the source code of a SAS program stored on the anonymous SAS FTP site and make a copy of it in a catalog entry of type source.

```
/*************************************************************/
/* Example of a FILENAME statement with the FTP option.      */
/*************************************************************/
filename copyin ftp 'missto0.sas'
  cd='/techsup/download/datastep/'
  user='anonymous' host='ftp.sas.com'
  pass= 'Your email address';
filename copyout catalog 'work.mycat.missto0.source';

data _null_;
  infile copyin sharebuffers;
  file copyout;
  input;
  put _infile_;
run;
```

THE URL ACCESS METHOD

You would like to read a web page directly into your SAS program? Nothing is easier thanks to the URL access method. URL stands for Universal Resource Locator and is an address that provides the route to a file (a web page for example) on the Internet. Therefore, when an address is specified, the corresponding web page will be read from that point on using normal SAS statements. Because most web pages on the Internet are encoded using HTML, your program will receive the actual HTML pages with all the tags that are embedded. Depending on what you intend to do, this may require the parsing of the web page to retain useful data. In the example below, we access the same file as in the case of the FTP example, but this time we use the HTTP protocol as implied with the "http: " prefix on the external file name provided. Note that the amount of information that needs to be provided is much less than with the FTP access method.

```
/*************************************************************/
/* Example of a FILENAME statement with the URL option.      */
/*************************************************************/
filename copyin url 'http://ftp.sas.com/techsup/download/datastep/missto0.sas';
filename copyout catalog 'work.mycat.missto0.source';

data _null_;
    infile copyin sharebuffers;
    file copyout;
    input;
    put _infile_;
run;
```

CONCLUSION – MORE DEVICES

As you can see the FILENAME statement is quite powerful. But there's more! Only a subset of the available options was covered in this paper. In the Windows environment, the following devices (and more!) are also supported:

- SOCKET: With the SOCKET access method, you can use SAS to communicate with another application over a TCP/IP socket in either client or server mode. The client and server applications can reside on the same machine or on different machines that are connected by a network

- DDE: Dynamic Data Exchange (DDE) is a method of dynamically exchanging information between Windows applications. DDE uses a client/server relationship to enable a client application to request information from a server application. SAS is always the client. In this role, SAS requests data from server applications, sends data to server applications, or sends commands to server applications.

- COMMPORT: to read data into SAS directly from the communications (serial) port on your machine.

- DRIVEMAP: to display information about the available hard drives (local and networked).

- PRINTER: to access a PRINTER.

- TERMINAL: to access the user's terminal.

SAS is known to provide a very flexible processing environment. By supporting so many different devices, the FILENAME statement in conjunction with the power of the DATA step is a fine example of this flexibility. This paper is a short summary of the capabilities of the FILENAME statement and the reader is encouraged to investigate further using the different resources listed in the References.

REFERENCES

SAS Institute Inc. 2004. *SAS 9.1 Language Reference: Dictionary*, Volume 2. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2004. *SAS 9.1 Companion for Windows*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2004. *SAS OnlineDoc 9.1.3*. Cary, NC: SAS Institute Inc.

Burlew, Michele M. 2002. *Reading External Data Files Using SAS®: Examples Handbook*. Cary, NC: SAS Institute Inc.

Chapman, David D. 2004. "Using SAS® Catalogs to Develop and Manage SAS® DATA Step Programs". *Proceedings of the Twenty-ninth Annual SAS Users Group International Conference*, Montreal, Canada, 051-29.

Davis, Michael. 2002. "Reading From Alternate Sources: What To Do When The Input Is Not a Flat File". *Proceedings of the Twenty-seventh SAS Users Group International*, Orlando, Florida, 006-27.

LeBouton , Kimberly J. 2002. "Smokin' With UNIX Pipes". *Proceedings of the Twenty-fifth SAS Users Group International*, Indianapolis, Indiana, 103-25.

Worden, Jeanina, and Philip Jones. 2004. "You've Got Mail – E-mailing Messages and Output Using SAS® EMAIL Engine". *Proceedings of the Twenty-ninth Annual SAS Users Group International Conference*, Montreal, Canada, 178-29.

Roper, Christopher A. 2001. "Accessing and Utilizing the Win32 API From SAS". *Proceedings of the Twenty-sixth Annual SAS Users Group International Conference*, Long Beach, California, 281-26.

ACKNOWLEDGMENTS

The author would like to thank the members of the SAS Technology Centre at Statistics Canada for providing many comments to improve this paper. Special thanks go to Pierre Lafrance for his continued support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Yves DeGuire
Statistics Canada
14 floor section O, R.H. Coats Building
100 Tunney's Pasture Driveway
Ottawa, Ontario, Canada, K1A 0T6
Phone: (613) 951-1282
Fax: (613) 951-0607
E-mail: yves.deguire@statcan.ca
Web: www.statcan.ca